Implementing New Clustering Algorithm for Cellular Subpopulations Identification

Prepared by

Lorraine Ramel
203352703
sk8er@campus.technion.ac.il

Nikita Dizhur
336492871
snikdizh@campus.technion.ac.il

Atalya Alon
2033852507
Atalya.alon@campus.technion.ac.il

Project in Artificial Intelligence
Winter 2017-2016
Technion

Table of contents

Introduction

Analyzing and understating cellular subpopulations behavior and their biological actions in a disease can play a critical role in understanding its mechanism, as well as discovering significant stratifying biological signatures within single cell datasets which can contribute to the development of diagnostic methods.

Single cell analysis technologies such as flow cytometry have enabled scientists to investigate in detail the role of specific cellular subpopulations and their diversity.
Flow cytometry is a laser-based biophysical technology, widely used for analyzing the expression of cell, characterizing and defining different cell types in a heterogeneous cell population, assessing the purity of isolated subpopulations and analyzing cell size and volume. Currently, it permits concurrent measurements of 12 to 17 parameters per cell, whereas Mass cytometry (CyTOF), a new platform of flow cytometry, increases the parameterization of single cell measurements to 40+ parameters per cell and produces rich datasets of cells.
Flow cytometry is a popular method, routinely used in the diagnosis of health disorder, especially cancer and has many applications in basic research or clinical trials, such as TIL/ACT.

TIL/ACT Background

TIL also known as ACT (Adoptive cell therapy) is an experimental therapy which involves removing some of a patient's own immune-system cells (white blood cells) from their tumors, growing them in the laboratory in large numbers, and then infusing the cultured cells back to the patient to destroy cancer cells. These cells are also called Tumor infiltrating lymphocytes (TIL). This type of treatment which known as immunotherapy, is systemic and effective in fighting advanced cancer, especially late stage melanoma[1].

TIL/ACT is mostly common in treating patients who have metastatic melanoma - which is a type of cancer that usually occurs in the skin, and appears to be unique among human cancers because of its ability to induce a significant numbers of lymphocytes with anti tumor activity (from now on we will call them markers) during the neutral course of tumor growth.
The ability to isolate and characterize these markers from patients with melanoma has thus enabled the identification and characterization of multiple melanoma antigen that can be the target of immunotherapy.

Historically, approaches to find stratifying cellular subsets have relied on manual identification, but today there are few methods that uses various strategies for identifying relevant subpopulations within the large size and complex datasets that CyTOF produces, such as clustering - a task of grouping a set of cells in such a way that cells in the same group are more similar to each other than to those in the other groups. However, most of these methods are often subjective and require prior knowledge of the biological system under investigation, and therefore do not scale well to higher-dimensional measurements.

Citrus (Cluster identification, characterization, and regression) is a clustering algorithm which was developed to tackle with the aforementioned task. It is data driven and based on unsupervised hierarchical clustering [2]. SPADE (Spanning-tree Progression Analysis of Density-normalized Events) is another method which is also based on organizing cells in hierarchy.

The main disadvantage of these methods is their hierarchical approach to clustering the cells data, which results in outputting many clusters with overlapping cell subpopulation sets that slows the analysis due to the increasing number of clusters, and therefore lower the ability to differentiate clusters that contribute more to the diagnosis from those that contribute less.

In this project, our data includes huge information of single-cell datasets belonging to melanoma patients in late stages. These melanoma patients had undergone TIL/ACT trials, where some of them responded positively and the others did not respond at all.

Our main challenge was to build a clustering method which takes the union of all the given detailed datasets and sort them into a number of groups so we later can identify sub populations of cells hidden in them. The constructed algorithm is robust, fast and based on the fact that some markers "contribute" more information to the process of defining sub populations of cells more than others. It outputs disjoint and non-hierarchical set of clusters as we wanted.

To demonstrate the clusters benefits, we conducted various experiments to explore how different values of the algorithm parameters influence the outcome of the analysis.
These experiments consist of building a classifier that is based on a decision-tree with features extracted from the obtained clusters. At last we discussed and compared our classifiers performances to Citrus classifier performance, which is established with Citrus clusters.

Solution Description and Procedures Used

Building The Clustering Algorithm

The clustering procedure we suggest simulates building a "Binary tree" in a preorder traversal. At the beginning, the root will contain thousands of initial cells that are sampled randomly from the patients' dataset of cells, and then we want to split the root into two separated nodes in such a way that cells in each node are more similar to each other than those in the other node.
The method of splitting will be based on a cell marker threshold value (reminder: the process of flow cytometry measures a number of parameters that characterize a single cell) and its calculation relies on the distribution of cells in the current node to be split. After dividing the initial data, we continue splitting the obtained nodes in the same way recursively, meaning each received node is split into two nodes and so on.
We notice that our splitting process basically separates different subpopulations of cells from each other, or in other words - we elucidate more and more subpopulations every time, thus the obtained leaves from the overall process will express our targeted clusters.
In general, the main algorithm is constructed in 3 main steps:

1. Sampling process over the patients' cells.
2. Performing a splitting method based on a pre-knowledge markers.
3. Performing a splitting based on a regular markers.

The sampling process

Our database stores thousands of cells for each patient. For this reason to prepare the datasets of cells for clustering, it is sufficient to perform a simple sampling process over each patient's dataset of cells: given a fixed K number, loop over the given patients and sample K cells randomly from their dataset of cells, then merge the sampled cells altogether into the root node.
Formally, given N patients denoted $H = \{h_1, h_2, \dots, h_n\}$, where $h_i.cells$ store the sampled cells of patient h_i. Therefore, $root = \cup_{i=1}^{N}(h_i.cells)\,, where\ \forall i \in [1, N], |h_i.cells| = K.$

Defining a splitting method

We already know that some markers (parameters) contribute more information to the task of defining sub populations more than others. Therefore we cluster the patients' cells based on these markers and specifically those who mostly feature the population of cells.
For seeking a good data partition in a certain node denoted as P, our focal point relies on finding a marker which gives a higher information gain when splitting the high dimensional data in the current node. To find such marker, initially we set thresholds values to the featuring markers to calculate their entropies, then we choose the marker with the highest entropy value over the data, and thus divide P to two nodes denoted as R and as L such that $P = R \cup L$ and $R \cap L = \emptyset$ hold.
Markers' thresholds are defined based on a combination of normal mixture model and k-means analysis on the distance distribution (following pseudo code). These thresholds separate the cells into two groups: R is designated to be the "Minus" population group, such that $\forall c \in R$ $c[M] > t_M$ where $c[M]$ is the value of marker M in cell c, t_M is the marker threshold, and L is designated to be the "Plus" population group such that $\forall c \in L, c[M] \leq t_M$ respectively.

Pseudo code - finding markers thresholds

Function: $CalculateThresholds(cells, markers)$:

0. $\lambda = [0.25,0.25,0.25,0.25]$ #initial value of mixing proportions
1. $\epsilon = 1e-05$, $k = 4$
2. $thresholds = \emptyset$
3. **foreach** m **in** $markers$:
   - 3.1 $mix = \mathbf{normalMixEm}(cells[m], \epsilon, \lambda, k)$ #initialize a normal mix analysis
   - 3.2 $km = \mathbf{kMeans}(mix.data, k = 2, centers = (\min(mix.data), \max(mix.data))$ #apply kMean analysis with 2 components
   - 3.3 $kmThresh = \mathbf{max}(mix.data \in first\ km\ component)$
   - 3.4 $gausThreshs = mix.thresholds$ #gaussian thresholds given by mix
   - 3.5 $gausIdx = \mathbf{max}(\mathbf{index}(gausThreshs \leq kmThresh))$ # max index of gaussian thresh
   - 3.6 **if** $gausIdx == |gausThreshes|$
     - 2.6.1 $thresholds = thresholds \cup \max(gausThreshs)$
   - 3.7 **else**
     - 3.7.1 $contained = gausThresh[gausIdx] < mix.data < gausThresh[gausIdx + 1]$
     - 3.7.2 $propUnderKmThresh = \frac{|contained < kmThresh|}{|contained|}$
     - 3.7.3 **if** $propUnderKmThresh < 0.5$
       - 3.7.3.1 $thresholds = thresholds \cup gausThreshs[gaudIdx]$
     - 3.7.4 **else**
       - 3.7.4.1 $thresholds = thresholds \cup gausThreshs[gaudIdx + 1]$
4. **return** $thresholds$

First split stage – based on pre knowledge markers

From the known biological point of view, some of these markers are important and donate more than others to the progress of finding well defined subpopulations, or in other words – there are special markers that give us a good separation of the data in a certain order.

We want to apply our pre knowledge before using the entropy split strategy by "preferring" some markers over the others, let's call them "initial markers" - as they are the first markers we use for building the sub tree.

To formalize the splitting process of this stage, we denote $t^n_{M^i_j}$ as the threshold in node n and which corresponds to the initial marker M^i_j where $M^i_j \in \{M^i_1, M^i_2 \dots M^i_t\} = M^i$. The root separation occurs in such a manner that $\forall c \in root$, if $c[M^i_1] > t^{root}_{M^i_1}$ then $c \in R$ otherwise, $c \in L$. In the same way, we split with M^i_2 the obtained nodes: R with $t^R_{M^i_2}$, and L with $t^L_{M^i_2}$ and then the descendants of R and L with M^i_3 and so on until M^i_t. (Figure 1).

Pseudo code - first stage of splitting

Function: $SplitByBias(node, markers)$:

0. **if** $markers = \emptyset$:
   - 0.1 **return** $\{node\}$
1. $m \leftarrow markers.getFirst()$
2. $t \leftarrow \mathbf{CalculateThresholds}(node, \{m\})$
3. $left \leftarrow cells\ from\ node\ where\ cell[m] \leq t$
4. $right \leftarrow cells\ from\ node\ where\ cell[m] > t$
5. $unusedMarkers \leftarrow markers \backslash m$
6. **return** $SplitByBias(left, unusedMarkers) \cup SplitByBias(right, unusedMarkers)$.

Second split stage – based on a marker's entropy

When finishing the first stage of splitting, we move to split the leaves of the current sub tree to get more groups of cells. The splitting method is switched to the main splitting strategy which is based on the marker's entropy. When iterating over the leaves, we want to select the marker that gives the best information gain, i.e. the best possible split - meaning the marker with the maximal entropy value (marker's entropy is dependent on the marker's thresholds).

We know that m markers of the overall markers significantly characterize the cells, including the initial markers. So let's denote these markers as $M = \{M_1, M_2, \dots, M_m\}$, and the leaves received from the previous procedure as $N = \{l_1, l_2, \dots, l_k\}$. Therefore, $\forall i \in [1, k]$ set markers' thresholds as $T = \{t_1, t_2, \dots, t_m\}$, and find markers' entropies $E = \{e_1, e_2, \dots, e_m\}$ respectively. Let's assume $e_j = MAX\{e_i(t_i)\}\ \forall i \in [m]$, then $M_j \notin M^i$ (not an initial marker) is the chosen marker for splitting the cells in leaf l_i.

Splitting l_i implemented such that $\forall c \in l_i, if\ c[M_j] > t_j$ then $c \in R_{l_i}$ otherwise $c \in L_{l_i}$, where t_j is the corresponding threshold to M_j. Much as the same as in the first stage of splitting.

Each node in N (leaves) is split recursively in the same process until we obtain our targeted clusters. We notice that prior to each split, the current markers' thresholds are calculated once again since it is dependent on the distribution of cells which differs between the leaves and thus entropies are calculated again as well. In addition, during the recursive track of splitting l_i, each marker is selected only once (Figure 1).

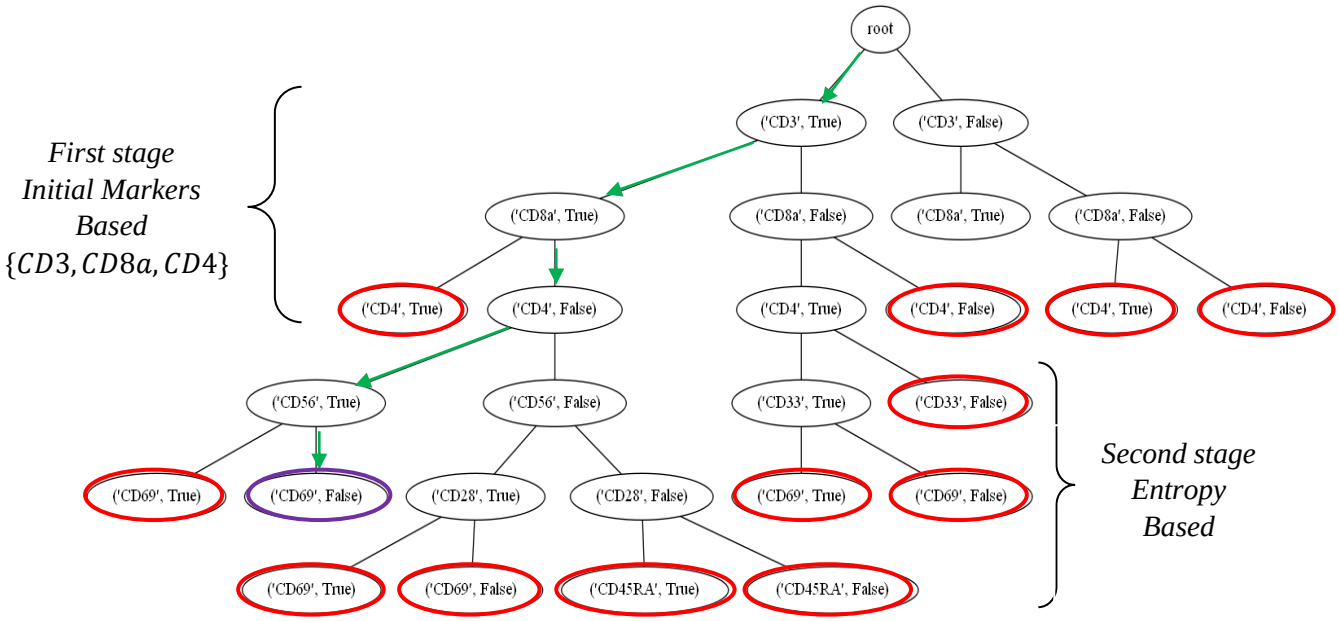


Figure 1. Mini tree illustration modeling the clustering algorithm

Every cell population (red circles) are characterized by only one sequence of markers expressions. To clarify this saying, lets denote the obtained clusters as $C = \{C_1, C_2, \dots, C_l\}$ and their tree paths as $P = \{P_1, P_2, \dots, P_l\}$ respectively, so $P_i = \{(M_j, s) \mid M_j \in M, s \in \{+, -\}\}$.

For example C_3 (Purple circle) is characterized by the path: $P_3 = \{(CD3, +), (CD8a, +), (CD4, -), (CD56, +), (CD69, -)\}$ (green path). It is important to notice that in every tree depth: $\forall R, L$ we have $R \cap L = \emptyset$. This indicates that our clusters don't overlap! Hence we achieve one of our targeted features towards building the clustering algorithm. In

addition to that, our clusters are only the leaves attained from the overall procedure. This means that our algorithm is not hierarchical, unlike Citrus for example.

Pseudo code - second stage of splitting

Function: $SplitByEntropy(nodes, minEnt, minCells, markers)$:

0. **if** $markers = \emptyset$:
   0.1 **return** $\{node\}$

1. **foreach** $node \in nodes$:
   1.1 **if** $node < minCells$
   1.1.1 **return** $\{node\}$

   1.2 $threshes \leftarrow$ **CalculateThresholds**$(node, markers)$
   1.3 $e \leftarrow maximal\ entropy\ over\ corresponding\ threshold\ (threshold \in threshes)$
   1.4 $m \leftarrow marker\ corresponding\ to\ chosen\ entropy$
   1.5 $t \leftarrow thresh\ corresponding\ to\ chosen\ marker$

   1.6 **if** $e < minEnt$**:**
   1.6.1 **return** $\{node\}$

   1.7 $left \leftarrow cells\ from\ node\ where\ cell[m] \leq t$
   1.8 $right \leftarrow cells\ from\ node\ where\ cell[m] > t$
   1.9 $unusedMarkers \leftarrow markers \backslash m$
   1.10 **return SplitByEntropy**$(left, minEnt, minCells, unusedMarkers) \cup$
   SplitByEntropy$(right, minEnt, minCells, unusedMarkers)$

Algorithm Parameters

We need to define two conditions for halting the splitting procedure to avoid data over fitting through the clustering process. In addition to the total cell parameter, our iterative clustering algorithm accepts two important parameters.

The first parameter is a lower bound on the minimal cell population in a node for splitting: if the cluster holds a number of cells in which it is higher than the minimal cells parameter, we allow splitting, otherwise we stop splitting and mark the node as a final cluster. It is important to set this parameter because it allows defining the number of minimal cells in a population of cells.

The second parameter is a bound on the minimal entropy of a marker. If the current elected marker's entropy in a node is less than the minimal entropy, we mark it as a final cluster.

This parameter is significant in the procedure of separating cells, thus we want to test its impact.

The two parameters are investigated during the second stage of the splitting procedure only, and prior to each split. That because during the first stage of splitting, we already know that the initial markers give us well defined subpopulations, so we would not want these parameters to restrict reaching them. Therefore, we apply these conditions when splitting by the regular markers only.

Pseudo code – The clustering algorithm

Algorithm: *Clusters' Compuation over patinets cells.*

Input:
A set of patients denoted $H = \{h_1, h_2, \dots, h_n\}$.
A set of markers denoted $M = \{M_1, M_2, \dots, M_m\}$.
An array of pre knowledge markers denoted $M^i = [M_1^i, M_2^i, \dots, M_t^i]$ *where* $M_j^i \in M$.
A sampling parameter denoted K
A minimal entropy denoted by $MinEnt$.
A minimal cells denoted by $MinCells$.

Output:
N *clusters, where* $N \in \mathbb{N}$.

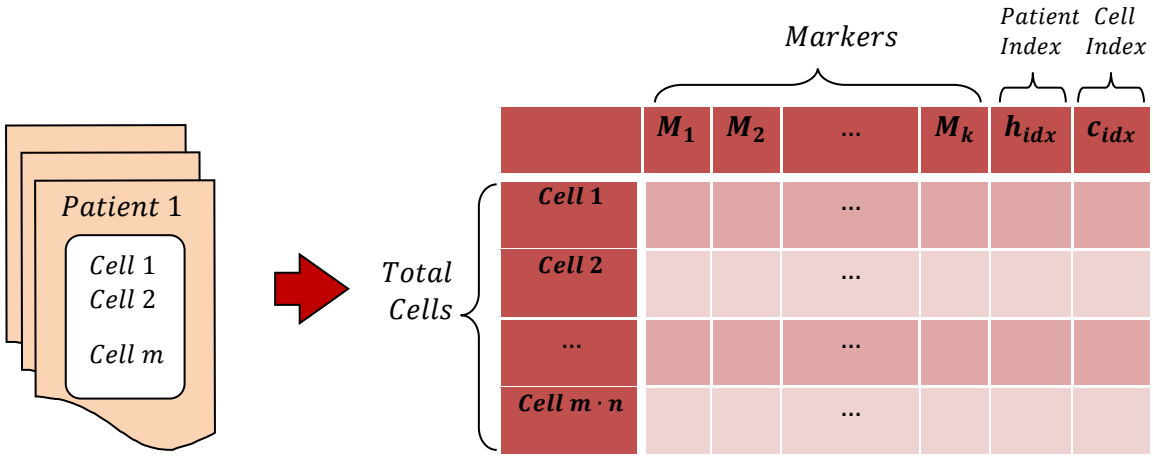
Function: $ComputeClusters(H, M, K, M^i, minEnt, minCells)$:

0. $root \leftarrow \emptyset$
1. **foreach** *patient* $h \in H$:
   1.1 $hCells \leftarrow$ *sample* K *random cells from* h *cells*
   1.2 $root \leftarrow root \cup hCells$
2. $biasNodes \leftarrow \mathbf{SplitByBias}(root, M^i)$
3. $clusters \leftarrow \mathbf{SplitByEntropy}(biasNodes, minEnt, minCells, M)$
4. **return** *clusters*

System Description

The whole system can be divided into 3 main levels:

1. Processing the CyTOF data level.
   The enormous data of cells that we have are basically an output of Mass Cytometry or CyTOF (Cytometry by time of flight) experiments which is a recent biophysical technology employed in characterizing cells and analyzing them. The CyTOF output is stored in FCS (Flow Cytometry Standard) files, so to be able to read them; we need to translate these data into readable formats (e.g. **Nnumpy** arrays).
   We used the **FCMeausrements** model under the **FlowCytometryTools** – a Python package used for performing flow cytometry analysis.



(i) Translating FCS files to Python tables

The patients' responses to TIL/ACT are annotated in a CSV file under "AI_Project/Classification/AnnotationFull.csv". Patient is marked as "Responder" if TIL/ACT was a success, otherwise "Non-Responder".
We map the patients' responses to binary values {0: "*Non Responder*", 1: "*Responder*"} to use them later as classification input to the learning algorithm (ii).

AnnotationFull.csv	
Patient 1	*Responder*
Patient 2	*Responder*
...	...
Patient n	*Non Responder*

Classifications	
Patient 1	1
Patient 2	1
...	...
Patient n	0

(ii) Translating patients respones to binary labels

2. Clustering system level

The processed data from the previous level is taken as an input to the clustering system to create the clusters that later are used as features.

As mentioned in the solution outline, distance thresholds are iteratively calculated for certain markers during the clustering method. For that we implemented a function programmed in R language, which uses some functionality from the **Mixtools** package – a library in R for analyzing finite mixture models.

Under **Mixtools** we used the **normalmixEm** model and the **kMean** clustering model to help us set the distance thresholds. Another model that was used is the **Rpy2** - a Python package for porting the R code to run in Python.

The following flowchart describes the steps of the clustering system part and the models' relations (Figure 2)

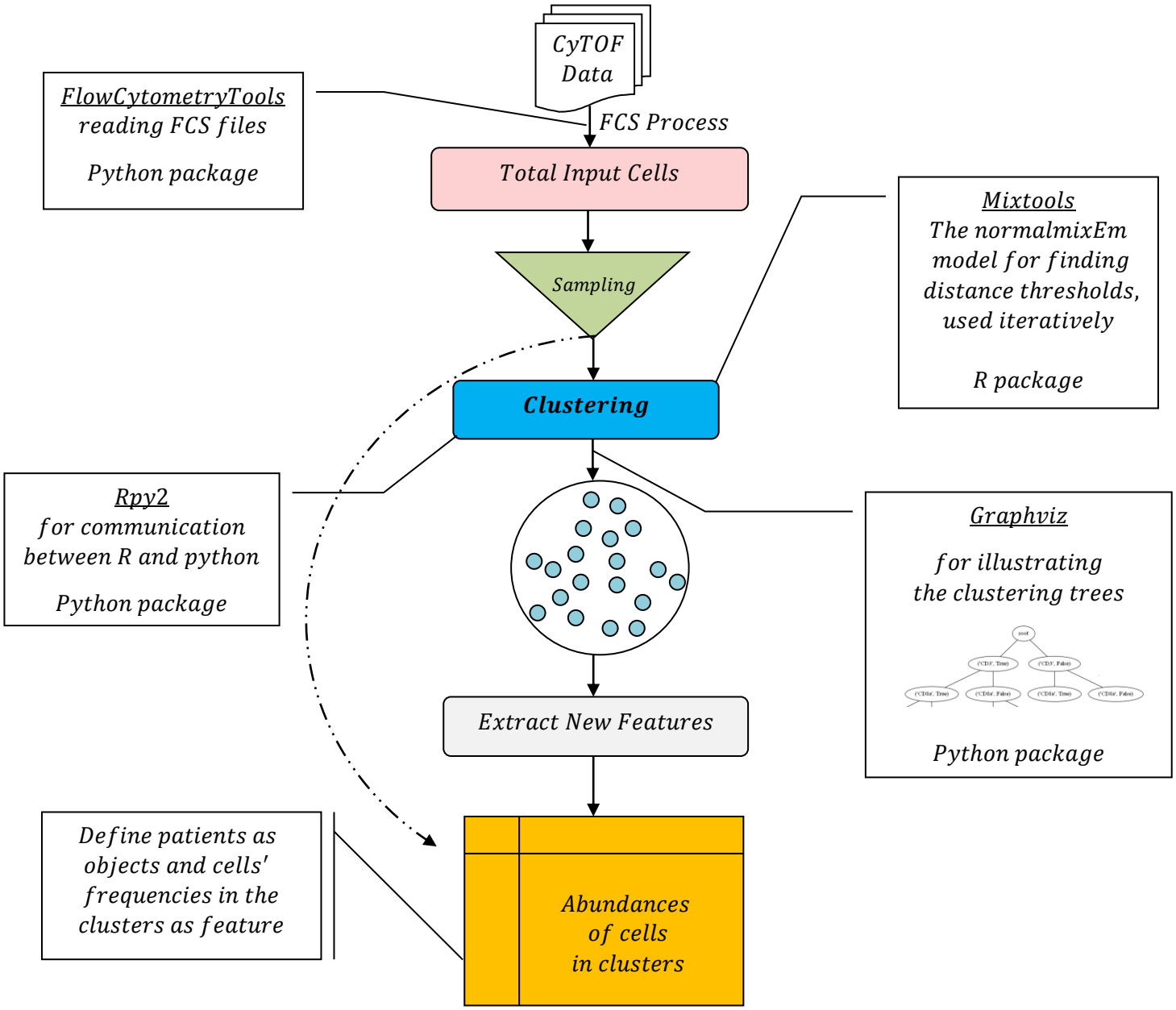


Figure 2. Flowchart of the clustering system

3. Classifier system level

The classifier model (Figure 3) uses the previous model output as an input, i.e. the cell abundances for every patient are used as a training data for generating a decision tree classifier.

We used sub models under **Sklearn** - a Python package for machine learning – the decision tree classifier model (for creating the classifier) and the model of cross validation (for evaluating the classifier performance).

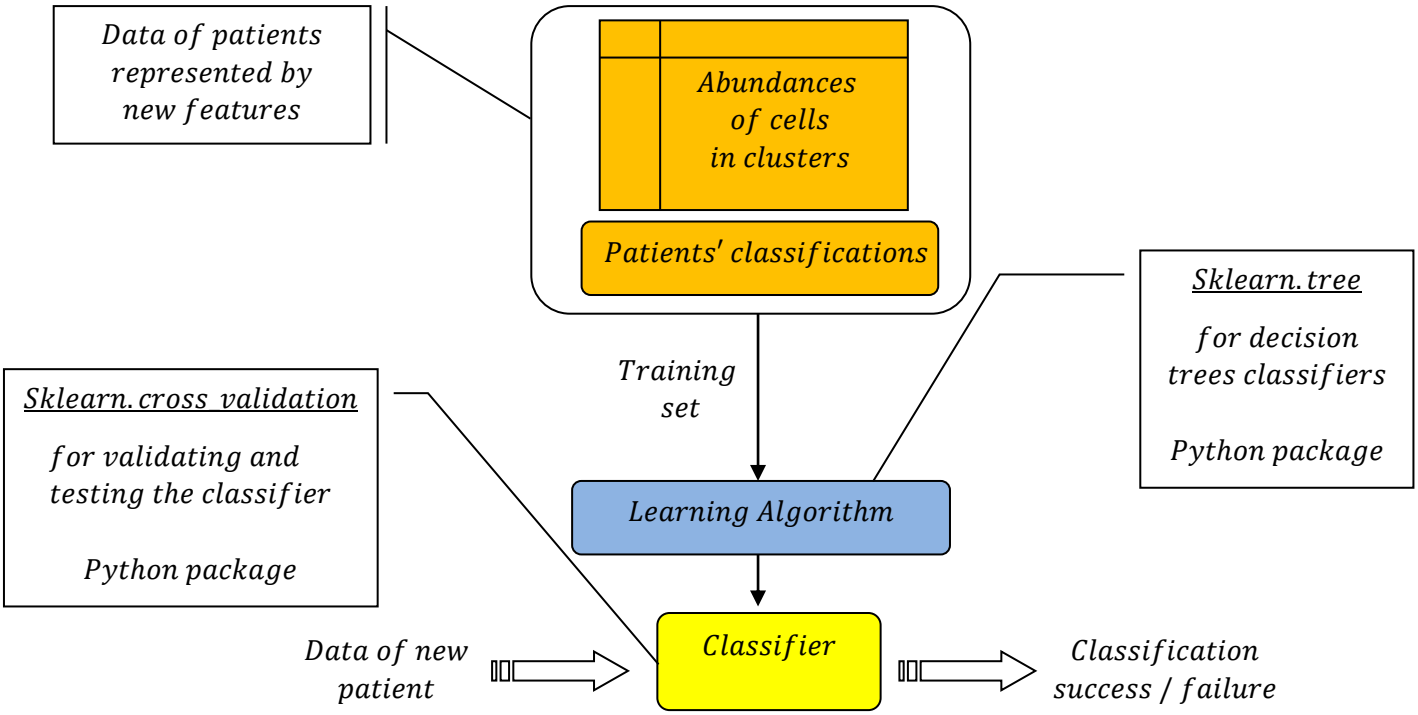


Figure 3. Flowchart of the classifier system

Methodology of Experiments

We created a new clustering method to help us improve the accuracy of deciding whether TIL/ACT would succeed on a given patient or whether it would fail. Therefore, to examine the clustering procedure, we conducted different kind of experiments that we divided into 3 stages:

1. Verification of results' consistency:
   Initially, we want to confirm that the clusters obtained from of the algorithm are consistent with the well known data received (Citrus data which was examined by scientists in the field). To achieve that, we compare the T-distributed Stochastic Neighbor Embedding (T-SNE) dimensionality reduction of our clusters against those of Citrus. In that way, we can check the ability of our algorithm to identify similar subpopulations and to find anomalies inside them with respect to each marker.

2. Exploring system variables:
   Next, we perform experiments with different system variables to understand their influence on the clusters' structure and prediction accuracy (at this point we determined system variables by the method of intelligent guessing).

3. Comparing clusters' "goodness":
   One of our clusters' benefits would be building a classifier with features extracted from the obtained clusters. At last, we compare the classifier prediction accuracy versus prediction accuracy of Citrus classifier.

To confirm consistency:

- We used a well-defined collection of cells (CyTOF single-cell datasets) derived from 40 patients of late stage melanoma, such that 21 of them responded positively to TIL/ACT, and the other 19 responded negatively.

- Cells of each patient were randomly sampled with the maximum of 50,000 cells.

- We used 15 markers out of 46 markers (Cell's parameters) in which they are biologically known to be important in identifying cell populations.
  These markers are: $M = \{CD3, CD8a, CD4, CD57, CD28, CD45RA, CD69, CD25, CD33, CD45RO, CD127, CD56, CD62L, CD45\}$, and the initial markers are $M^i = \{CD3, CD8a, CD4\}$.

- For system parameters we set 20,000 for minimal cells' amount in cluster and 0.1 for minimal entropy. Those values were chosen according to existing literature and by the fact that 20,000 is roughly 1% of the overall amount of sampled cells.

- Next we applied T-SNE to receive two-dimensional visualizations of the clustered data, described as follows:
  1. First we characterized each cluster C by a high dimensional vector, denoted as V_c.
  2. V_c is a vector of all the means of all the markers values in C, .i.e. if we denote $V_c = (m_1, m_2, \dots, m_{15})$ then $m_i = Mean(\{c[M_i] | \forall c \in C\})$ where c[M_i] is the value of marker M_i in cell c.
  3. After featuring the clusters, we perform T-SNE to reduce the high dimensional data (vectors) to be able to visualize them in two dimensions and then observe the clusters.

4. To understand the markers' expressions in the clusters we demonstrate a single visualization for every marker in which each cluster is colored based on the marker expression intensity (more in the experiments section).

We configured T-SNE parameters to preserve the structural properties of the data, i.e. clusters with more similarity will be closer to each-other while less similar clusters will reside farther away one from another.

- Finally, we examined how well we reflected relations between the subpopulations (and inside them) and to preserve overall structure of the data.

Then to test the system parameters:

1. We intend to explore different values of the algorithm parameters: 1.minimal entropy values and 2. minimal amounts of cells in clusters, in order to determine clusters' structure and "talkativeness" - how much they can contribute in dividing cells to different and consistent categories.
   For minimal cells' amount in cluster we use:
   a. 20,000 cells
   b. 100,000 cells

   It roughly constitutes 1% and 5% of the overall cell population respectively.
   For minimal entropy we use:
   a. 0.05
   b. 0.1
   c. 0.3
   d. 0.5
2. We want to investigate how changing the amount of examined patients in database can affect the prediction accuracy rate and to understand the correlation between the number of new patients added and prediction accuracy improvement. Therefore, we conduct experiments with different k-values in k-fold cross validation method to find optimal size of the train set.
   For fold sizes we use:
   a. 2
   b. 4
   c. 5
   d. 8
   e. 10
   f. 20

By using the model of decision tree classifiers we would be able to predict the patient's responsiveness to TIL/ACT.
The following steps describe the process of building the classifier:

1. Define the patients as objects and use the clusters set $C = \{C_1, C_2, \dots, C_l\}$ from the clustering algorithm to extract the features.
2. Assume $o_k =< v_1, v_2, \dots, v_l >$ is the features vector designated to patient h_k and define it as follows:

$$\forall i \in [1, l]\ v_i = \frac{|cells\ of\ h_k\ in\ C_i|}{|cells\ in\ C_i|} \cdot 100$$

   I.e. v_i is the percentage of cells belonging to patient k in cluster C_i.
3. Denote the objects set as $X = \{o_{h_k} \mid o_{h_k} =< v_1, v_2, \dots, v_l > \forall h_k \in H\}$
4. Denote the objects labels as $Y = \{l_k \mid l_k \in \{0,1\}\ 0\text{:"non-responder", }1\text{:"responder"}\}$.
5. Train a decision tree classifier given data X and labels Y.
6. Use the model of stratified k-fold to evaluate the classifier's performance

Experiments

Consistency verification: our clusters vs. Citrus

Let's discuss and compare some cell populations emitted from citrus and our clustering algorithm. To do so, let's observe the two clusters sets in 2D T-SNE visualizations:

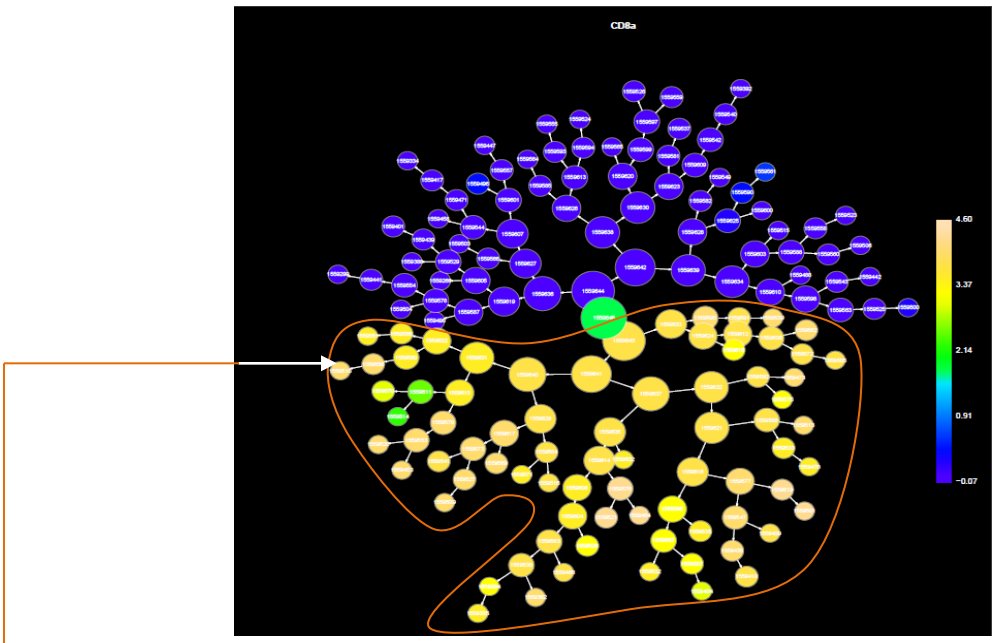


Figure 4. Visualization of CD8a with Citrus

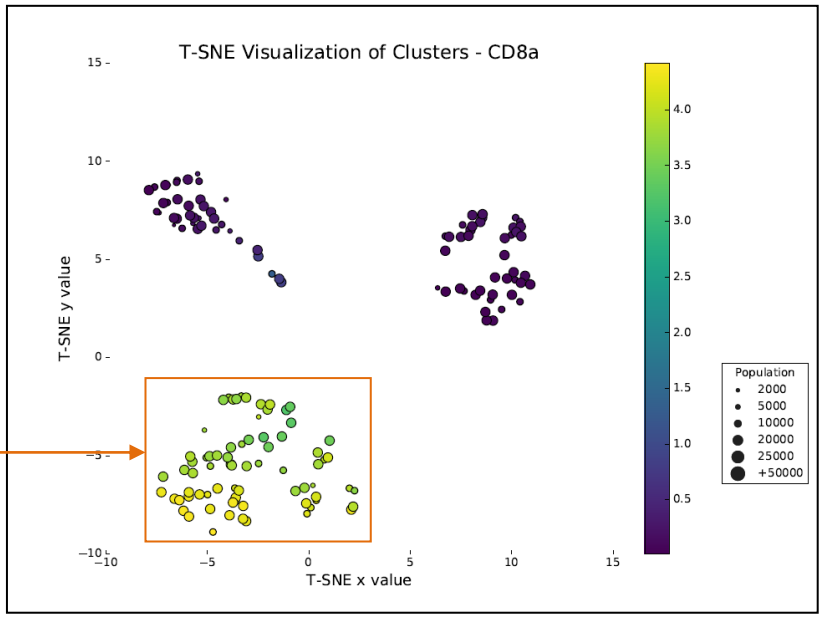


Figure 5. Visualization of CD8a with our clusters

The visualizations describe the distribution of CD8a marker with Citrus clusters (Figure 4) and our clusters (Figure 5), with sampling size 50000, minimum entropy 0.1 and minimum cells 20000.

The colored dots represent the clusters and the color bar demonstrates the marker intensity range. Now it is easy to notice there are some independent cell populations: both marked groups express clusters with a high value of CD8a accordingly in the graphs, while the other unmarked groups express cell populations with a low value of CD8a.

Let's observe another visualization of another initial marker – CD4:

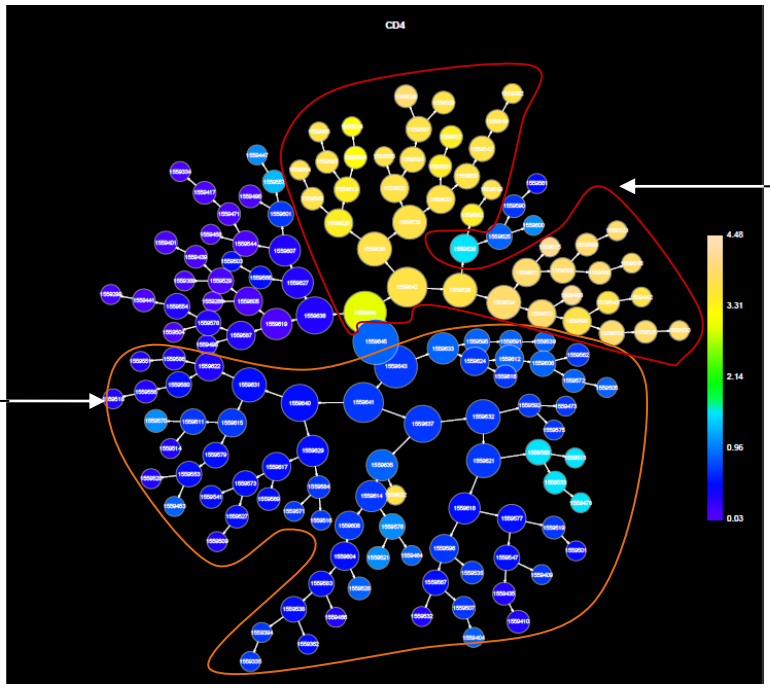
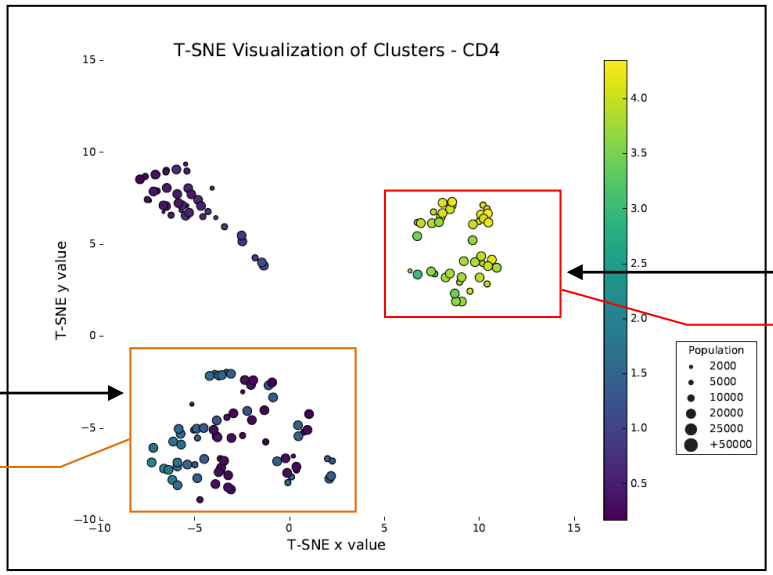


Figure 6. visualization of CD4 marker with Citrus



(i) The previous population that showed high value of CD8a, now shows low value of CD4, which also can be seen in the visualization of citrus.

(ii) This Population shows high value of CD4. we've already seen that it shows low value of CD8a, and it is also compatable with citrus visualization.

Figure 7. visualization of CD4 marker with our clusters

Now we can see that the left bottom group (i) which expressed a high value of CD8a, now expresses a low value of CD4. The upper right group (ii) expressed a low values of CD8a, yet

higher values of CD4. Thus, we recognize two groups of cell populations from our clusters (Figure 7) where they match Citrus groups. The unmarked group expresses low values of both CD8a and CD4 and it also matches the unmarked clusters in Citrus, as expected (Figure 6).
It is sufficient to note – obviously there is one small subpopulation in Citrus's clusters who is anomaly whereas our clusters don't show it – it is an interesting case which might require further investigation.
For certainty, let's continue and research more markers:

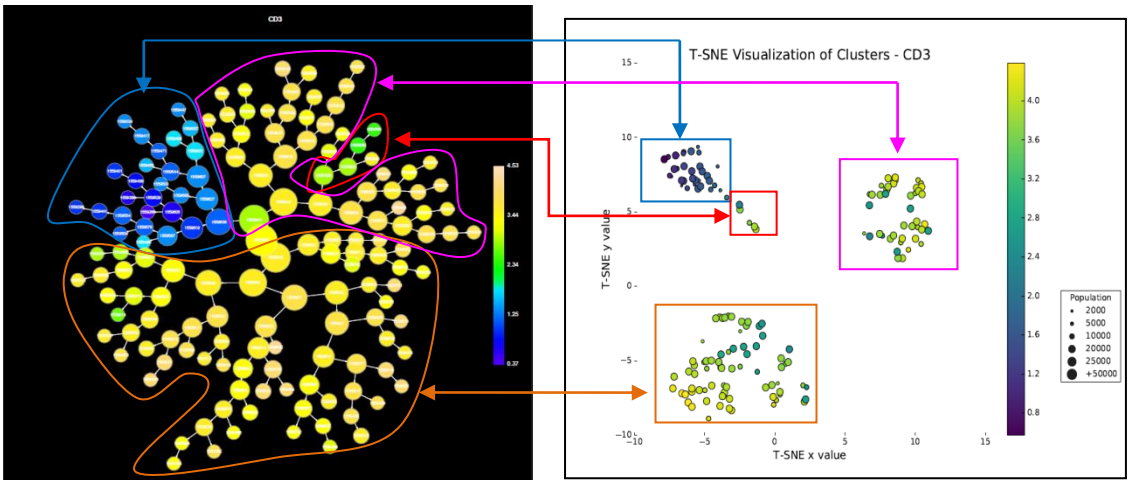


Figure 8. visualization of CD3 marker

With CD3 (third initial marker) we can identify another sub population of cells.
Since we have recognized the big populations (orange and pink squares, Figure 8) let's try to observe the upper left group (red and blue squares, Figure 8). We already know it expresses low values of CD8a and CD4, but we can see there is a small sub population in which there is a number of clusters with high values of CD3 (red square) and the rest shows significantly lower values of CD3. This sub population is visible and marked too in Citrus's visualization.

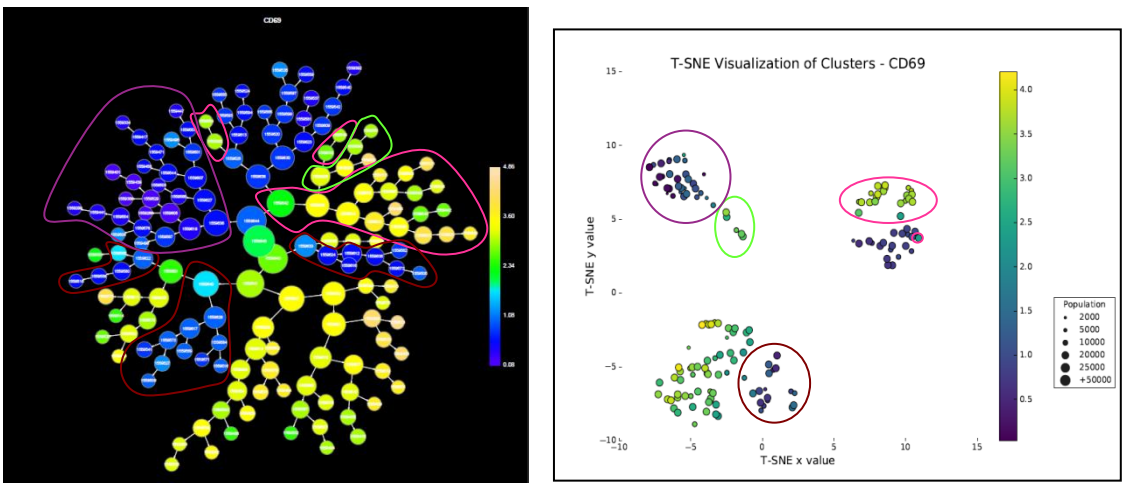


Figure 9. visualization of CD69 marker

In the same manner, we can recognize more and more sub populations that are consistent with Citrus results, as we intended to prove in this section. The more we investigate through markers' visualizations, the more we can see a match between our subpopulations and the Citrus ones. E.g. in Figure 9, more sub populations are detected when observing CD69 marker.

Experimenting different system variables

We have implemented 8 different experiments with all the combination of system variables. The following table sums up all the combinations of the main experiments variables:

Experiment	Sampling size	Minimal Entropy	Minimal cells
1	50,000	0.05	100,000
2	50,000	0.05	20,000
3	50,000	0.1	100,000
4	50,000	0.1	20,000
5	50,000	0.3	100,000
6	50,000	0.3	20,000
7	50,000	0.5	100,000
8	50,000	0.5	20,000

Table 1. Experiments variables of the clustering algorithm

Each experiment is a single run of the clustering algorithm in which it outputs a group of sets that represent the clusters. After creating the clusters we process a table of the cells abundances in clusters for all patients and then train a classifier over the data. To estimate the classifiers' performance, we perform the stratified k-fold cross validation technique with different k values. The following graphs describe the results of the classifiers' accuracy rate with different k-folds:

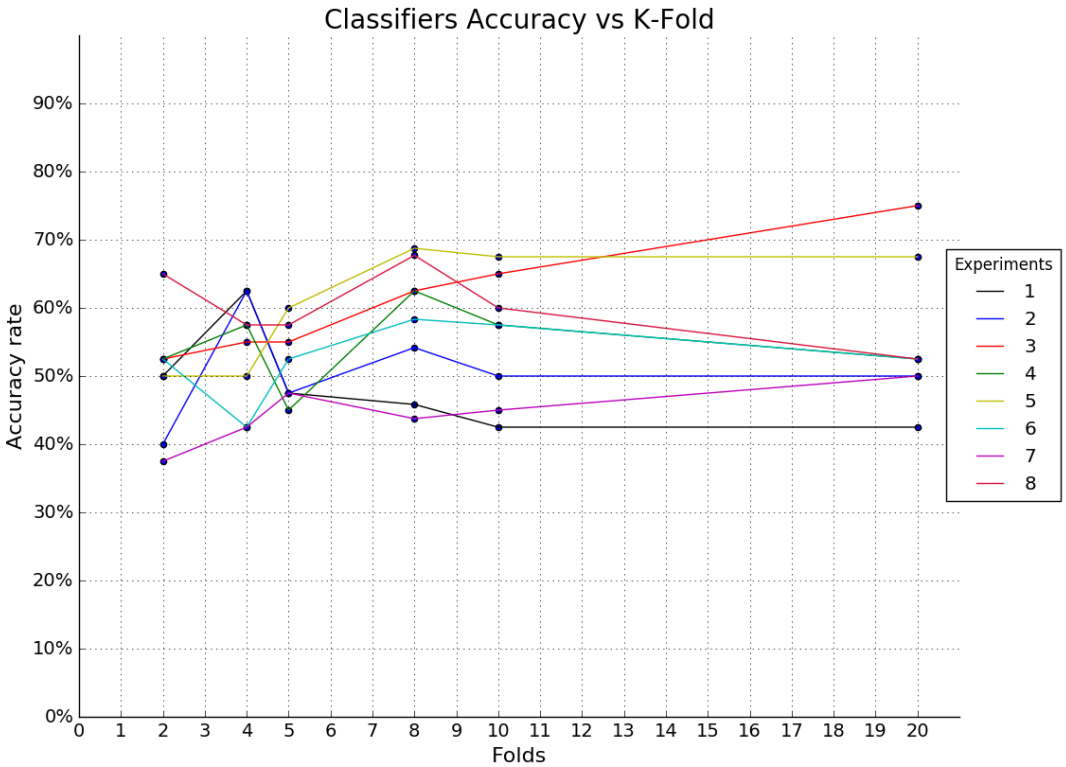


Figure 10. Classifier accuracy per fold

Now let's summarzie all experiments results:

Experiment	Sampling size	Minimal Entropy	Minimal cells	Classifier Accuracy					
				2	4	5	8	10	20
1	50,000	0.05	100,000	0.500	0.625	0.475	0.458	0.425	0.425
2	50,000	0.05	20,000	0.400	0.625	0.475	0.541	0.500	0.500
3	50,000	0.1	100,000	0.525	0.550	0.550	0.625	0.650	0.75
4	50,000	0.1	20,000	0.525	0.575	0.450	0.625	0.575	0.525
5	50,000	0.3	100,000	0.500	0.500	0.600	0.687	0.675	0.675
6	50,000	0.3	20,000	0.525	0.425	0.525	0.583	0.575	0.525
7	50,000	0.5	100,000	0.375	0.425	0.475	0.437	0.450	0.500
8	50,000	0.5	20,000	0.650	0.575	0.575	0.677	0.600	0.525

Table 2. Experiments results

Cursory glance

One can notice that experiments 3, 5 and 8 relatively express some good results among all the other experiments, while the rest are less impressive.
In general, when talking about what affects building the clusters, we should observe changing the values of minimal entropy and minimal cells. Lower entropies values or lower values of minimal cells permit more splitting during the clustering algorthim in which it can reveal hidden subpopulations of cells. However, an excessive division might also result in data overfitting. Higher entropies values or higher minimal cells values would restrict widen the clustering tree since the splitting stops eariler, thus not exposing certain subpopulations of cells – if they exist, and then possibly causing the opposite case - data underfitting.
The prominent experiments (Table 2) include an "opposite" combination of these two different parameters. For example, experiment 3 parameters include high minimal cells value (100000) but low minimal entropy (0.1), and experiment 8 parameters include low minimal cells value (20000) but high miniml entropy value (0.5).
These experiments' results imply that both parameters are affected by each other and work simultaneously when building the clustering tree. They both impact the quality of the eventual clusters. In the end, it is resonable to assume that optimal results will most likely appear among experiments which include choosing lower minimal entropies values with higher minimal cells values, or the other case - choosing higher minimal entropies values with lower minimal cells values.

Lets observe the experiments results with the fold which gives highest average accuracy value across the different parameters:

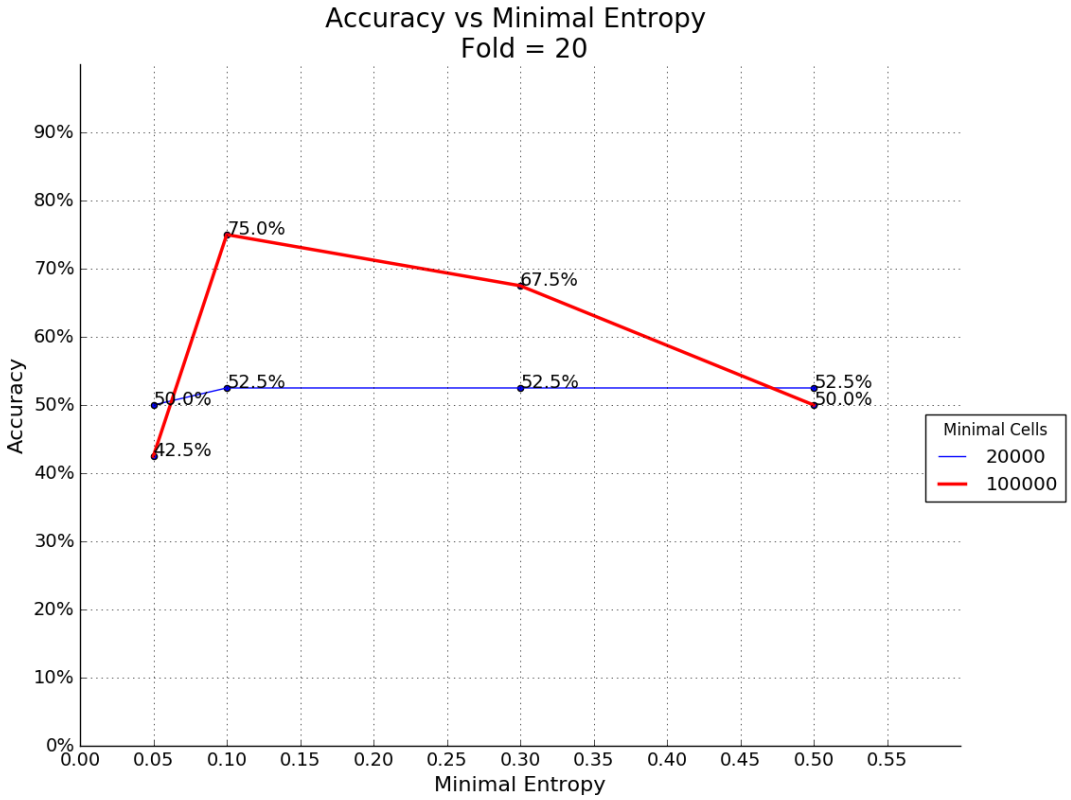


Figure 11. Classifier accuracy with 20-Fold

CASE 1 - 100000 minimal cells

Minimal entropy values lower than 0.1 show a drastic decrease of the accuracy average value. Although our minimal cells variable is high and according to the previous discussion, we expect optimal accuracies with higher minimal cells value and lower minimal entropy. Yet we received very low accuracy. Therefore, our conclusion is that 0.05 is too low for minimal entropy and we should seek higher minimal entropies between $[0.1 - \epsilon, 0.1 + \epsilon]$ with 100000 minimal cells. This significant gap might also imply that our data begins to over fit with entropies ≤ 0.1.
Data over fitting is possible since lower entropies allow more splits when building the clustering tree as we discussed before, and that results in creating many small clusters which are basically groups of small subpopulations. In such cases, the process may not necessarily contribute to identification of relevant subset of cells, but rather making the process of finding similar data is difficult.
Minimal entropy values over 0.3 tend to show a decrease in prediction rates too. As expected, higher entropies with higher minimal cells together do not contribute to splitting and widen the clustering tree. Thus, our clusters are few and have a huge population of cells which is not defined well. This manner expresses the other extreme case, which is data under fitting and it is reflected well with minimal entropy 0.5.
The highest accuracies are obtained over the segement [0.1,0.3] of minimal entropy values, particularly 0.1 is the best minimal entropy among our results.
Future researches would conduct new experiments with minimal entropy values that lie between 0.1 and 0.3, and preferably values around 0.1 to find more optimal accuracy.

Now let's observe the other case - 20000 minimal cells.

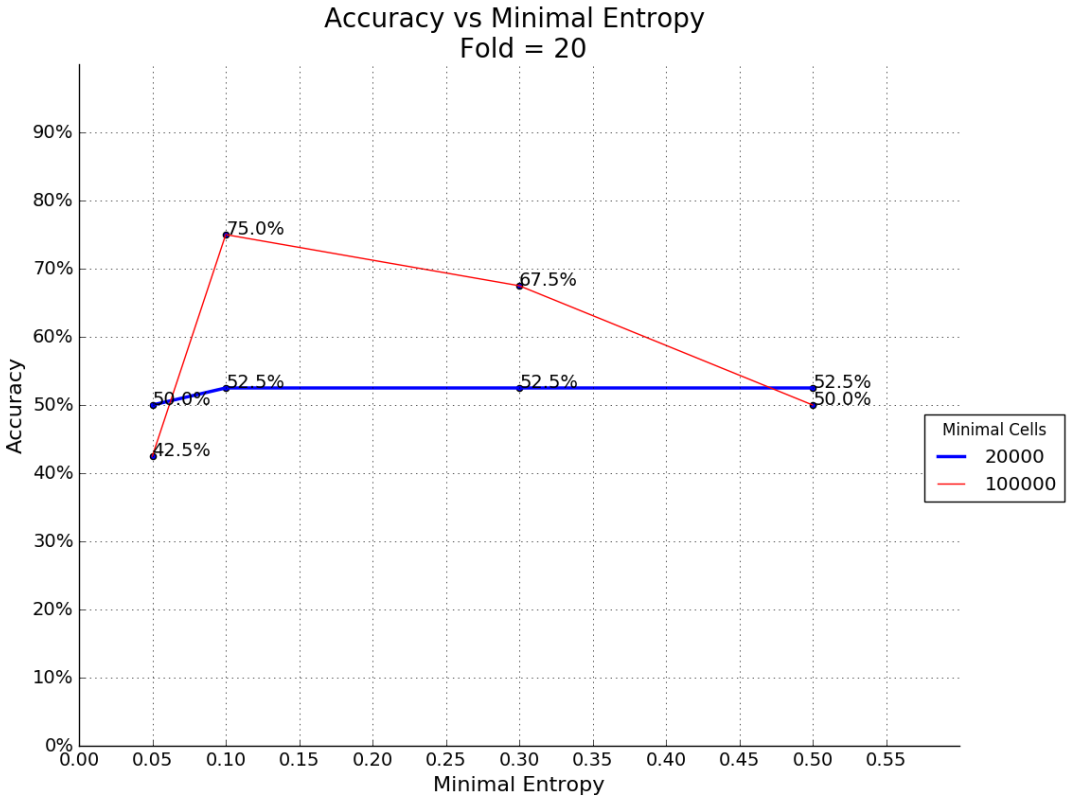


Figure 12. Classifier accuracy with 20-Fold

CASE 2 - 20000 minimal cells

As seen in the graph, the average accuracies express a stable manner with minimal entropy values ≥ 0.1, while they begin to decrease with minimal entropy values lower than 0.1 – which is likely caused by data overfitting.

Since now our minimal cell variable is much lower than 100000, it is interesting to observe higher minimal entropy values according to our initial discussion. But because our experiments did not include minimal entropy values higher than 0.5, thus we lack information to determine the accuracy rate manner in this case. However, we highly expect a rise in the accuracy rate at some point. Therefore, a future reasearch would conduct experiments with minimal entropy values ≥ 0.5 to understand what are the optimal values of minimal entropy with 20000 minimal cells.

The following table sums up the best accuracy rate of each experiment classifier:

Experiment	Sampling size	Minimal Entropy	Minimal cells	k-fold	Accuracy
1	50,000	0.05	100,000	4	0.625
2	50,000	0.05	20,000	4	0.625
3	50,000	0.1	100,000	20	0.750
4	50,000	0.1	20,000	8	0.625
5	50,000	0.3	100,000	8	0.687
6	50,000	0.3	20,000	8	0.583
7	50,000	0.5	100,000	20	0.500
8	50,000	0.5	20,000	8	0.677

Table 3. Classifiers best accuracy with fold

The first two experiments demonstrate an interesting results and case:
One can notice that we received the same accuracy rate, though minimal cells' number differs, while other parameters have the same value. It can point to an interesting conclusion: neither 100,000, nor 20,000 are optimal parameter values for this algorithm with respect to other parameters. Apparently the best minimal number of cells lies somewhere in between and thus, the case of 20,000 may manifests data overfitting, while the case of 100,000 may express data underfitting. At last, finding the optimal value for this parameter demands further experimentation.

Accuracy Comparison: our clusters' abundances vs. Citrus's

Given Citrus' data, the cells' abundances in Citrus clusters, or as we called them in other words: patients' cells' frequencies. We generate a decision tree classifier and train it over Citrus data such as before, meaning the classifier features are extracted from Citrus clusters.
We evaluate the classifier performance by performing a stratified cross validation with all the k-fold values we used before.
The following graph describes the performance of Citrus classifier:

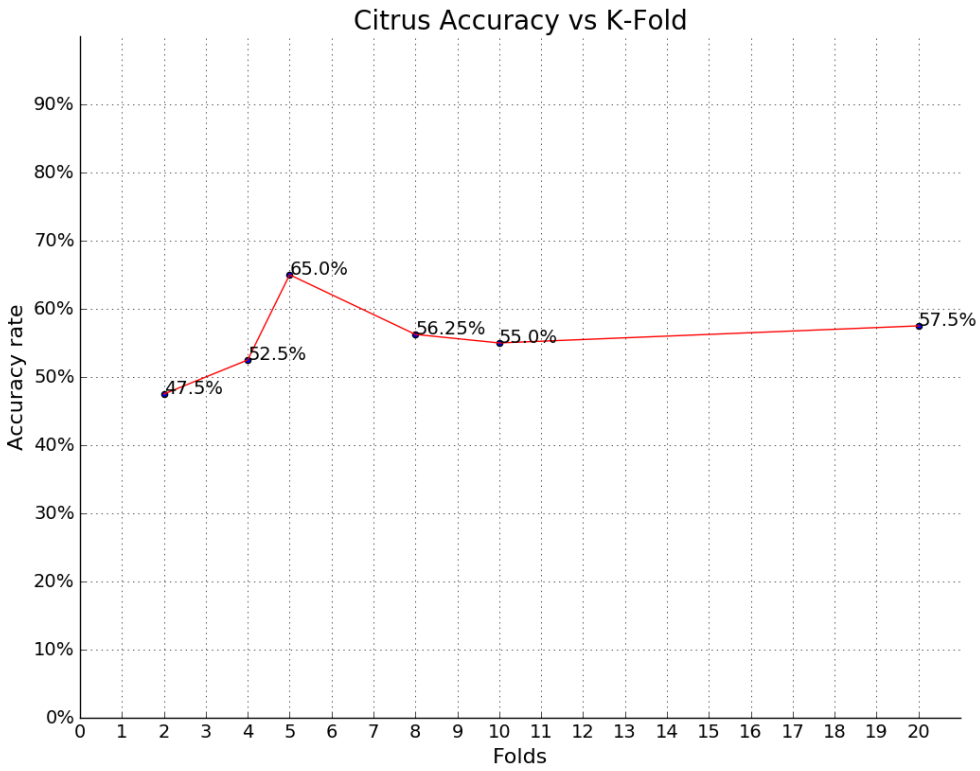


Figure 13

A quick observation over the results of Citrus, we can tell in general that we surpassed Citrus optimal result; our clusters achieved 75% rate success, while Citrus achieved only 65% at the most. Most of our experiments results beat Citrus' result when looking at the same K-fold. Moreover, we can see despite the fact that sometimes the average Citrus' results are better than those of our less successful experiments, but with the right parameter tuning our algorithm easily outperforms Citrus by 5% to 6%.
It's sufficient to note that the best result Citrus has shown (65% success rate) is about 10% lower than the best result that was received by our algorithm (75% success rate).
We believe this was achieved due to the advantages of the non-hierarchical clustering (our clustering algorithm) over the hierarchical one (Citrus) and hence creating more meaningful clusters with more expressive power.

Conclusions

In this project, we have presented and tested a new clustering algorithm for cellular subpopulations identification based on the data received from CyTOF (single cell analysis), with the purpose to use those clusters to predict the results of TIL/ACT treatment on a given patient.

The idea behind the algorithm was to eliminate the drawbacks that other clustering algorithms have, such as Citrus, by using a non-hierarchical approach to cluster creation. It allows us to create more meaningful clusters, while reducing their overall amount. Also, we have used biased biological knowledge to determine the importance and the order of some markers.

In this study we have shown, using a variety of tests, that the algorithm handles the given task well enough, at the same time allowing us to adjust some of its parameters with respect to the current data. Resulting groups of clusters confirm to the expectations from the biological point of view and do not contradict the existing models, but on the contrary – strive to complement them. Nevertheless, it is clear that for the further improvement of the algorithm's results an additional experimentation is needed.

With all that said, the next step of the research would be applying our algorithm to a different batch of patients, in order to compare and to validate our results in different environments and conditions. It is also important to investigate correlations between different batches of patients and to what extent system that has been trained on a specific group of patients can be applicable to a different one that had been examined under different conditions and circumstances.

Finally, we believe that in current studies and researches that involves exploring the behavior of cellular subpopulations, our clustering algorithm has a great potential and can be used to complement and one day even replace existing clustering methods in this field.

References

1. Steven A Rosenberg and Mark E Dudley: **Adoptive cell therapy for the treatment of patients with metastatic melanoma.**

2. Robert V. Bruggner, Bernd Bodenmiller, David L. Dill, Robert J. Tibshirani and Garry P. Nolan: **Automated identification of stratifying signatures in cellular subpopulations.**

3. Peng Qiu, Erin F. Simonds, Sean C. Bendall, Kenneth D. Gibbs Jr., Robert V. Bruggner, Michael D. Linderman, Karen Sachs, Garry P. Nolan, and Sylvia K. Plevritis: **Extracting a Cellular Hierarchy from High-dimensional Cytometry Data with SPADE**.

4. Nima Aghaeepour, Greg Finak, The FlowCAP Consortium, The DREAM Consortium, Holger Hoos, Tim R Mosmann, Ryan Brinkman, Raphael Gottardo & Richard H Scheuermann: **Critical assessment of automated flow cytometry data analysis techniques.**